Unidad 02: Conceptos Fundamentales de Algoritmia

2

Contenidos analíticos

Parte I: Del Universo real al universo computacional

Unidad 2: Conceptos Fundamentales de Algoritmia
Del problema real a su implementación en computadoras. Análisis y comprensión de problemas de información planteados en lenguajes natural. Representaciones graficas para la formalización y resolución estratégica de problemas computables de información. Concepto de algoritmo. Refinamientos sucesivos. Lexico y algoritmo. Características, propiedades y eficiencia de los algoritmos. Estructura de los algoritmos. Proceso computacional. Definición de paradigmas de programación. Profundización del paradigma imperativo, modular y estructurado. Concepto de Computación, Informática, Sistemas. Tipos de problemas: problemas computacionales. Definición de asignación, sentencia, declaración, dato, información, conocimiento, lenguajes de programación. Partes de un programa, Abstracción, Modularización, Especificación

Del problema real a su solución por computadoras

Analizando un problema del mundo real se llega a la *modelización* del problema por medio de la *abstracción*.

A partir del modelo se debe elaborar el análisis de la solución como sistema, esto significa la descomposición en módulos. Estos módulos deben tener una función bien definida.
La modularización es muy importante y no solo se refiere a los procesos a cumplir, sino también a la distribución de los datos de entrada, salida y los datos intermedios necesarios para alcanzar la solución.

Estudio de los datos del problema.
Cada módulo debe tener un proceso de refinamiento para expresar su solución en forma ordenada, lo que llevara a la construcción del algoritmo correspondiente.
A partir de los algoritmos se pueden escribir y probar programas en un lenguaje determinado y con un conjunto de datos significativos.

Etapas de resolución de problemas con computadoras.

1. Análisis del problema: en su contexto del mundo real.
2. Diseño de la solución: Lo primero es la modularización del problema, es decir la descomposición en partes con funciones bien definidas y datos propios estableciendo la comunicación entre los módulos.
3. Especificación del algoritmo: La elección adecuada del algoritmo para la función de cada modulo es vital para la eficiencia posterior.
4. Escritura del programa: Un algoritmo es una especificación simbólica que debe convertirse en un programa real sobre un lenguaje de programación concreto.
5. Verificación: una vez escrito el programa en un lenguaje real y depurado los errores sintácticos se debe verificar que su ejecución conduzca al resultado deseado con datos representativos del problema real.

Análisis de problemas planteados en lenguaje natural

Enunciado en lenguaje natural

Dado un conjunto de valores enteros, calcular e imprimir: cantidad de ceros, promedio de positivos, sumatoria de negativos

Aquí dice "dado", esto significa que se deben evaluar un conjunto de datos, lo que se requiere para poder resolverlo desde la perspectiva de la programación donde están estos datos, de donde se obtienen. Esto puede ser por asignación interna o externa de entrada.
Aquí se determina el tipo de dato que se evaluara, dice entero, lo que no esta claro es cual es el tamaño del conjunto. Es posible que un enunciado lo determine, como veremos mas abajo, o que quien de la respuesta limite el conjunto, esto es un ejemplo de definición o precondición, por ejemplo se evaluaran 50 datos.
Lo otro que debe establecerse, para poder definir cuanto estacio de memoria debe reservarse es ver que hacer con los datos. Puede ser que con cada dato haya que, como en este caso saber como es respecto de cero, por lo cual no es nebesario conservarlo después de la evaluacion, en este caso habrá que determinar un único identificados que se va modificando durante el proceso.
Distinto es si por ejemplo se dice dado un conjunto de valores enteros mostrarlos ordenados según algún criterio, aquí se necesita el valor ingresado y se lo debe mantener para hacer la evaluacion posterior.
Luego dice "calcular e imprimir", esto debe comprenderse como evaluar según se solicita (calcular) y mostrar los resultados "imprimir" en el dispositivo que se seleccione

Resolver el problema para los siguientes lotes de datos:
10 valores enteros.
N valores, donde el valor de N debe ser leído previamente.
El conjunto de valores termina con un valor igual al anterior.

Esto solamente determina el tamaño del lote a efectos de seleccionar la composición iterativa mas adecuada
Se puede pensar que "calcular" esta vinculado con una evaluacion en función de las características del dato evaluado, lo que correspondería a un análisis de caso, por otro lado cuando se dice "dado un conjunto….." se puede vincular al tamaño del lote lo que sugiere la elección de alguna composición iterativa

La forma correcta de solución algorítmica es la de refinamientos sucesivos, es decir buscar primero una solucion general, a nivel estratégico y luego comenzar a refinar la solución

Ejemplo el enunciado propuesto para un lote de N valores (segunda opcion)

```
// solución a nivel estratégico
Declarar los identificadores
Leer (N)  obtener el valor N que indicara el tamaño del lote
Repetir N veces
   Leer(valor) obtener cada uno de los valores a evaluar
   Evaluar el numero(valor, ceros, positivos, negativos) Analizar el subconjunto

Mostrar Resultados
```

El refinamiento se debe centrar en la declaración estricta de los identificadores y la resolución de cada uno de los modulos evaluar el numero y mostrar los resultados

*Dado un conjunto de triángulos representados por sus lados **L1**, **L2** y **L3** que finaliza con un triángulo con un lado nulo, determinar e imprimir la cantidad de triángulos equiláteros, isósceles y escalenos.*

Aquí se pueden pensar varias alternativas

1. El enunciado garantiza que los valores ingresados forman un triangulo, si la respuesta es si se puede establecer eso como pre condición, si la respuesta es no es responsabilidad de quien desarrolla determinar eso
    a. Los valores deben ser mayores a cero
    b. Cada lado debe ser menor a la suma de los otros dos
2. Aquí no esta claro el tipo de dato, quien desarrolla puede determinarlo
3. Dice que finaliza con uno de los lados nulos, es decir con esto se puede determinar la composición iterativa

*Dados dos valores **N** y determinar e imprimir cuantos múltiplos de **M** hay dentro del conjunto 1 a **N**.*

Aquí se precisa cuantos son los valores dados, simplemente dos, dado que se buscan múltiplos es razonable pensar que esos valores son enteros.
Existen algunas cosas que requieren mayor definición

1. Esos valores como deben ser respecto a cero
2. ¿Se admiten dos valores iguales?
3. ¿N debe ser mayor a N?

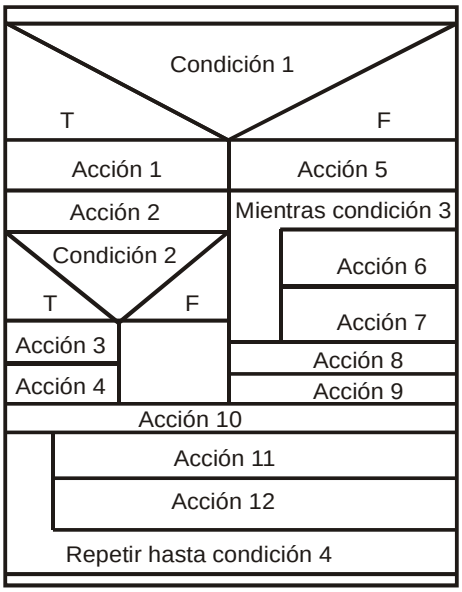
Cual es la consigna determinar la cantidad de múltiplos o también decir cuales son. El enunciado parece lo primero

Dado un valor **M**, determinar e imprimir los **M** primeros múltiplos de 3 que no lo sean de 5, dentro del conjunto de números naturales.

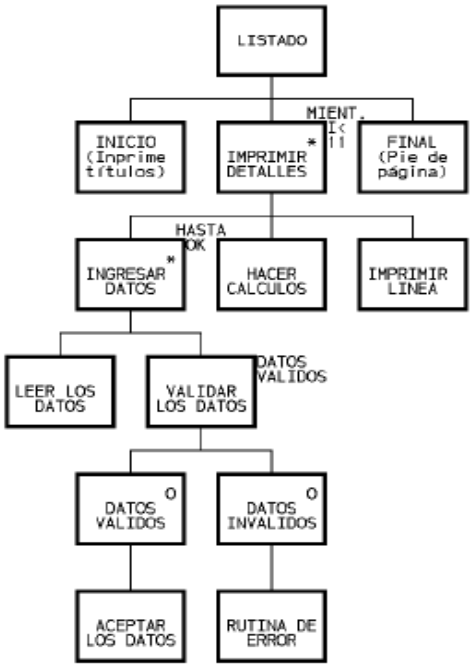
Aquí solo se ingresa un valor y esta claramente definido al conjunto que pertenece

Representaciones graficas para la formalización y resolución estratégica de problemas computables de información

Diagrama de Nassi-Sneiderman

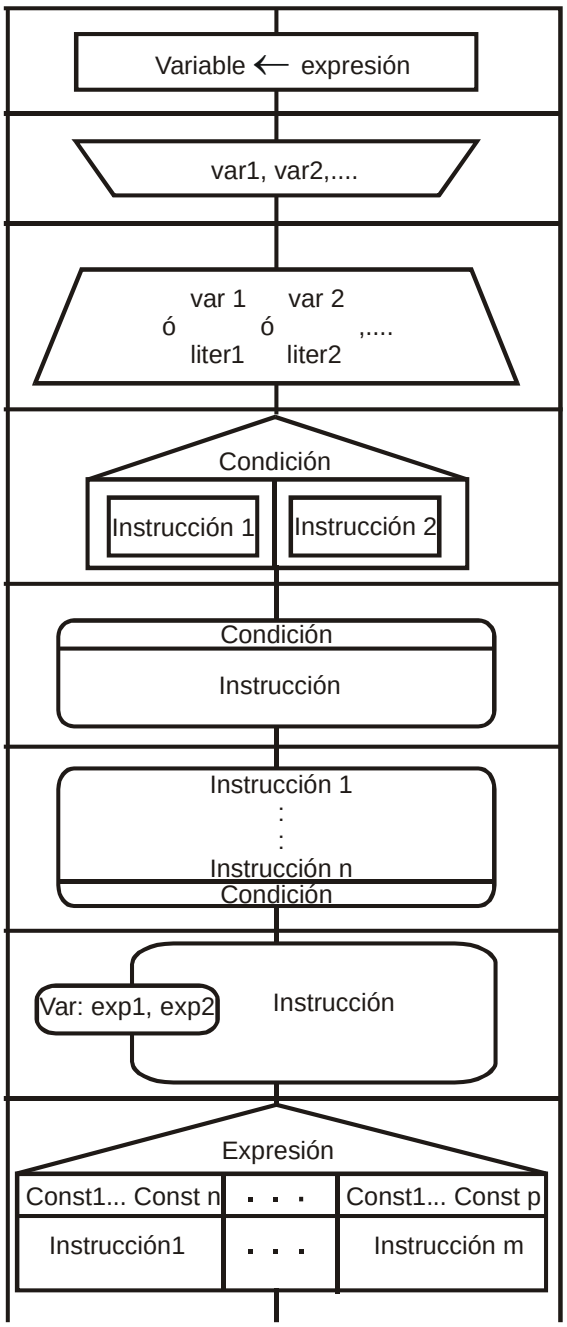


Diagramas de Jackson

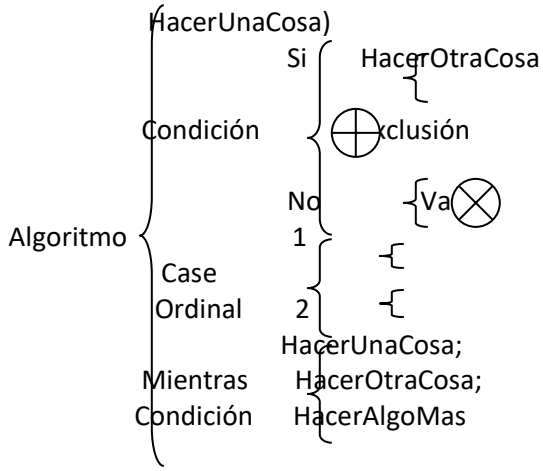


Diagramas de Lindsay.

DIAGRAMA DE DETALLE



Llaves de Warniel



Notación Algoritmica

LEXICO {Léxico Global del algoritmo}
{Declaración de tipos, constantes, variables y acciones}
Acción 1
PRE {Precondición de la acción 1}
POS {Poscondición de la acción 1}
LEXICO {Léxico local, propio de la acción 1}
Declaraciones locales
ALGORITMO {Implementación de la acción 1}
{Secuencia de instrucciones de la acción 1}
FIN {Fin implementación algoritmo de la acción 1}

ALGORITMO
PRE {Precondición del algoritmo principal}
POS {Poscondición del algoritmo principal}
{Secuencia de instrucciones del algoritmo principal}
FIN {Fin del algoritmo principal}

Equivalencias entre notación algorítmica y lenguajes de programación

CONDICIONAL

Formato	C

SI Condicion
ENTONCES
S
SI_NO
R
FIN_SI
	If (expresion)
	S;
	else
	R;

Formato	C

SEGÚN expr
V1 : S1
V2 : S2
EN_OTRO_CASO : Sn
FIN_SEGÚN
	switch (selector) {
	case etiqueta:S; break;
	.....
	default: R;
	}

ITERACION

Formato	C
Mientras Cond. Hacer S FIN_MIENTRAS	while(expresion) S;

Formato	
REPETIR S MIENTRA Cond	do S; while(expresion)

Formto	C
PARA i [Vi..Vf] HACER S FIN_PARA	for(i=0;i<vf;i++) S;

Estilos de Indentación

Recomendacion de estilos de indentacion para desarrollos más claros y legibles.(Ing. J. M. Sola)

Estilo The One True Brace Style

```
while( SeaVerdad() ) {
      HacerUnaCosa();
      HacerOtraCosa();
}
HacerUnaUltimaCosaMas();
```

BSD/Allman.

```
while( SeaVerdad() )
{
      HacerUnaCosa();
      HacerOtraCosa();
}
HacerUnaUltimaCosaMas();
```

Estilo Whitesmiths

```
while( SeaVerdad() )
      {
      HacerUnaCosa();
      HacerOtraCosa();
      }
HacerUnaUltimaCosaMas();
```

Estilo GNU

```
while( SeaVerdad() )
{
      HacerUnaCosa();
      HacerOtraCosa();
}
HacerUnaUltimaCosaMas();
```

Estilo Pico

```
while( SeaVerdad()
{     HacerUnaCosa();
      HacerOtraCosa(); }
HacerUnaUltimaCosaMas();
```

Estilo Banner

```
while( SeaVerdad() ) {
HacerUnaCosa();
HacerOtraCosa();
}
HacerUnaUltimaCosaMas();
```

Definiciones

Abstracción

Proceso de análisis del mundo real con el propósito de interpretar los aspectos esenciales de un problema y expresarlo en términos precisos.

Modelizacion

Abstraer un problema del mundo real y simplificar su expresión, tratando de encontrar los aspectos principales que se pueden resolver, requerimientos, los datos que se han de procesar y el contexto del problema.

Precondición

Información conocida como verdadera antes de iniciar el programa.

Poscondición

Información que debiera ser verdadera al cumplir un programa, si se cumple adecuadamente el requerimiento pedido.

Especificación

Proceso de analizar problemas del mundo real y determinar en forma clara y concreta el objetivo que se desea. Especificar un problema significa establecer en forma univoca el contexto, las precondiciones el resultado esperado, del cual se derivan las poscondiciones.

Algoritmo

El termino algoritmo es en honor del matemático árabe del siglo IX, *Abu Jafar Mohamed ibn Musa Al Khowârizmî*. Refiere conjunto de reglas, ordenadas de forma lógica, finito y preciso para la solución de un problema, con utilización o no de un computador.

En la actualidad al término se lo vincula fuertemente con la programación, como paso previo a la realización de un programa de computación aunque en realidad es una metodología de resolución presente en muchas de las actividades que se desarrolla a lo largo de la vida.

Desde los primeros años de escuela se trabaja con algoritmos, en especial en el campo de las matemáticas. Los métodos utilizados para sumar, restar, multiplicar y dividir son algoritmos que cumplen perfectamente las características de precisión, finitud, definición y eficiencia.

Para que el algoritmo pueda ser fácilmente traducido a un lenguaje de programación y luego ser ejecutado la especificación debe ser clara, precisa, que pueda ser interpretada con precisión y corresponda a pocas acciones, si esto no ocurre será necesario acudir a desarrollar un mayor nivel de refinamiento.

La utilización de refinamientos sucesivos es lo que permite alcanzar la solución modular que se propone.

Diseño modular, entonces, es la aplicación del criterio de refinamientos sucesivos, partiendo de un plan de acción, determinando que hacer, por aplicación de los conocimientos estratégicos de resolución pasando luego al como hacerlo con los conocimientos tácticos para la realización del algoritmo.

La programación de algoritmos representa un caso de resolución de problemas que requiere representación mental del mundo real, adaptación para tener una solución computable y criterio para elegir una alternativa eficiente de implementación.

Cuando se analiza un problema, particularmente de programación, y éste es difícil de describir, el plan de acción recomendable para alcanzar la solución es comenzar trazando un esbozo de las

formas más gruesas, para que sirvan de andamio a las demás; aunque algunas de ellas se deban cambiar posteriormente. Después, se agregan los detalles, (obteniéndose el algoritmo refinado), para dotar a estos esqueletos de una estructura más realista.
Durante la tarea de integración final, se descartan aquellas primeras ideas provisionales que ya no encajan en la solución. Por lo que, hasta que no se haya visto el conjunto global es imposible encontrarle sentido a ninguna de las partes por sí solas.
Siempre es mejor explicar un misterio en términos de lo que se conoce, pero cuando esto resulta difícil de hacer, se debe elegir entre seguir tratando de aplicar las antiguas teorías, o de descartarlas y probar con otras nuevas. Siguiendo este análisis, se define como reduccionistas a aquellas personas que prefieren trabajar sobre la base de ideas existentes, y como renovadores a los que les gusta impulsar nuevas hipótesis. En programación debe encontrarse un equilibrio entre ambas posturas.
La programación como toda actividad que requiere creatividad necesita que se produzca un salto mental que se puede sintetizar como señala David Perkins en:

1. Larga búsqueda, se requiere esfuerzo en analizar y buscar.
2. Escaso avance aparente: el salto mental sobreviene tras un avance que parece escaso o no muy evidente, pero sobreviene.
3. Acontecimiento desencadenante: El típico proceso de hacer clic comienza con un acontecimiento que lo desencadena.
4. Chasquido cognitivo: De pronto aparece la solución la que sobreviene con rapidez que hace que las piezas encajen con precisión, aun cuando sea necesario todavía ajustar algunos detalles. Pero la idea generadora apareció.
5. Transformación. Este avance nos va modificando nuestro mundo mental.

En síntesis, la practica del salto de pensamiento requiere en primer lugar de buscar analogías, en segundo lugar juegan un papel importante las conexiones lógicas, formulación de una pregunta crucial ocupa un papel decisivo. El repertorio de acciones tras el salto del pensamiento se expande para incluir no solo la analogía sino una extrapolación lógica y la formulación de la pregunta adecuada
Para encontrar una solución muchas veces se necesita desplazarse bastante por el entorno adecuado. Conforme a esto Thomas Edison declaro que la invención significa 99% de transpiración y 1% de inspiración, en contraposición con Platón que sostenía que las soluciones aparecen por inspiración divina.
Muchos problemas son razonables, cabe razonarlos paso a paso para alcanzar la solución. Otros son irrazonables no se prestan a un a reflexión por etapas.

Definición
Algoritmo
Especificación rigurosa (debe expresarse en forma univoca) de la secuencia de pasos, instrucciones, a realizar sobre un autómata para alcanzar un resultado deseado en un tiempo finito. Esto último supone que el algoritmo empieza y termina, en el caso de los que no son de tiempo finito (ej. Sistemas en tiempo real) deben ser de número finito de instrucciones.

En definitiva un algoritmo es una especificación ordenada de la solución a un problema de la vida real. Son el fundamento de la programación de computadores en el paradigma de programación imperativo.
Bajo este paradigma desarrollar un programa significa indicarle al computador, con precisión, sin ambigüedad y en un lenguaje que este pueda entender, todos y cada uno de los pasos que debe ejecutar para lograr el objetivo propuesto.

Previo a la traducción en un lenguaje de programación es necesario poder entender el problema, conocer las pre condiciones, establecer cual debe ser la pos condición, o aquello que debe ser cierto al finalizar la ejecución del algoritmo, en definitiva entender claramente *Que* es lo que se debe hacer para luego avanzar en *Como* hacerlo. Aquí debe utilizarse todas las herramientas al alcance de la mano para el desarrollo del algoritmo como paso previo a la solución del problema por el computador.
Existen varias técnicas para representar formalmente un algoritmo, una descriptiva llamada pseudo código, y otras graficas como los diagrama de flujo, diagrama Nassi Sneiderman, Diagramas de Lindsay, diagramas de Jackson, entre otros, en este caso se presentara una notación algorítmica similar a la presentada por Piere Scholl en el texto Esquemas algorítmicos fundamentales: Secuencia e iteración.

Definición
Algoritmo:
Secuencia finita de instrucciones, reglas o pasos que describen en forma precisa las operaciones que una computadora debe realizar para llevar a cabo una tarea en tiempo finito [Knuth, 1968].
Descripción de un esquema de comportamiento expresado mediante un repertorio finito de acciones y de informaciones elementales, identificadas, bien comprendidas y realizables a priori. Este repertorio se denomina léxico[Scholl, 1988].

Esta formado por reglas, pasos e instrucciones.
Las reglas especifican operaciones.
La computadora es el agente ejecutor.
La secuencia de reglas y la duración de la ejecución son finitas.

Características de un algoritmo

Un algoritmo debe tener al menos las siguientes características:

1. **Ser preciso**: esto significa que las operaciones o pasos del algoritmo deben desarrollarse en un orden estricto, ya que el desarrollo de cada paso debe obedecer a un orden lógico.
2. **Ser definido**. Ya que en el área de programación, el algoritmo es el paso previo fundamental para desarrollar un programa, es necesario tener en cuenta que el computador solo desarrollará las tareas programadas y con los datos suministrados; es decir, no puede improvisar y tampoco inventará o adivinará el dato que necesite para realizar un proceso. Por eso, el algoritmo debe estar plenamente definido; esto es, que cuantas veces se ejecute, el resultado depende estrictamente de los datos suministrados. Si se ejecuta con un mismo conjunto de datos de entrada, el resultado deberá ser siempre el mismo.
3. **Ser finito**: esta característica implica que el número de pasos de un algoritmo, por grande y complicado que sea el problema que soluciona, debe ser limitado. Todo algoritmo, sin importar el número de pasos que incluya, debe llegar a un final. Para hacer evidente esta característica, en la representación de un algoritmo siempre se incluyen los pasos inicio y fin.
4. **Presentación formal**: para que el algoritmo sea entendido por cualquier persona interesada es necesario que se exprese en alguna de las formas comúnmente aceptadas; pues, si se describe de cualquier forma puede no ser muy útil ya que solo lo entenderá quien lo diseñó. Las formas de presentación de algoritmos son: el pseudo código, diagrama de flujo y diagramas de Nassi/Schneiderman, entre otras. En esta publicación se propondrá una

notación algorítmica y se darán las equivalencias entre la propuesta y las existentes y también con las sentencias de los lenguajes de programación, en particular Pascal y C.

5. **Corrección**: el algoritmo debe ser correcto, es decir debe satisfacer la necesidad o solucionar el problema para el cual fue diseñado. Para garantizar que el algoritmo logre el objetivo, es necesario ponerlo a prueba; a esto se le llama verificación o prueba de escritorio.
6. **Eficiencia**: hablar de eficiencia o complejidad de un algoritmo es evaluar los recursos de cómputo que requiere para almacenar datos y para ejecutar operaciones frente al beneficio que ofrece. En cuanto menos recursos requiere será más eficiente el algoritmo.

La vida cotidiana está llena de soluciones algorítmicas, algunas de ellas son tan comunes que no se requiere pensar en los pasos que incluye la solución. La mayoría de las actividades que se realizan diariamente están compuestas por tareas más simples que se ejecutan en un orden determinado, lo cual genera un algoritmo. Muchos de los procedimientos utilizados para desarrollar tareas cotidianas son algorítmicos, sin embargo, esto no significa que todo lo que se hace está determinado por un algoritmo.

El primer paso en el diseño de un algoritmo es conocer la temática a tratar, el segundo será pensar en las actividades a realizar y el orden en que deben ejecutarse para lograr el objetivo, el tercero y no menos importante es la presentación formal.

Propiedades de los algoritmos

1. Especificación precisa de la entrada: El algoritmo debe dejar claro el número y tipo de datos de entrada y las condiciones iniciales que deben cumplir esos valores de entrada para conseguir que las operaciones tengan éxito.
2. Especificación precisa de cada instrucción: cada etapa del algoritmo debe estar definida con precisión, no debe haber ambigüedades sobre las acciones que se deben ejecutar en cada momento.
3. Un algoritmo debe ser exacto y correcto: Un algoritmo se espera que resuelva un problema y se debe poder demostrar que eso ocurre. Si las condiciones de entrada se cumplen y se ejecutan todos los pasos el algoritmo entonces debe producir la salida deseada.
4. Un algoritmo debe tener etapas bien definidas y concretas, un número finito de pasos, debe terminar y debe estar claro la tarea que el algoritmo debe ejecutar.
5. Debe ser fácil de entender, codificar y depurar.
6. Debe hacer uso eficiente de los recursos de la computadora

Finitud: en longitud y duración.
Precisión: Determinar sin ambigüedad las operaciones que se deben ejecutar.
Efectividad: las reglas pueden ejecutarse sin el ordenador obteniéndose el mismo resultado.
Generalidad: Resolver una clase de problema y no un problema particular.
Entradas y salidas: puede tener varias entradas pero una sola salida, el resultado que se debe obtener.

Eficiencia de un algoritmo

Se pueden tener varias soluciones algorítmicas para un mismo problema, sin embargo el uso de recursos y la complejidad para cada una de las soluciones puede ser muy diferente.

La eficiencia puede definirse como una métrica de calidad de los algoritmos asociada con la utilización optima de los recursos del sistema de cómputo donde se ejecutara el algoritmo, su

claridad y el menor grado de complejidad que se pueda alcanzar. *Hacer todo tan simple como se pueda, no más (Albert Einstein).*

La eficiencia como factor espacio temporal debe estar estrechamente relacionada con la buena calidad, el funcionamiento y la facilidad del mantenimiento.

Medidas de eficiencia para N = 10.0000

Eficiencia		Iteraciones	Tiempo estimado
Logarítmica	$Log_2 N$	14	Microsegundos
Lineal	N	10.000	0.1 segundo
Logarítmica lineal	$N * Log_2 N$	140.000	2 segundos
Cuadrática	N^2	10.000^2	15-20 minutos
Poli nómica	N^k	10.000^K	Horas
Exponencial	2^N	$2^{10.000}$	Inmedible

Complejidades más comunes

1. Complejidad constante: se expresa como O(1). Se encuentra en algoritmos sin ciclos, por ejemplo en un intercambio de variables.
2. Complejidad logarítmica: Es una complejidad eficiente, la búsqueda binaria tiene esta complejidad.
3. Complejidad lineal: se encuentra en los ciclos simples.
4. Complejidad logarítmica lineal: Los mejores algoritmos de ordenamiento tienen esta complejidad.
5. Complejidad cuadrática: Aparece en el manejo de matrices de dos dimensiones, generalmente con dos ciclos anidados.
6. Complejidad cúbica: Aparece en el manejo de matrices de tres dimensiones, generalmente con tres ciclos anidados.
7. Complejidad exponencial: es la complejidad de algoritmos recursivos.

Léxico y algoritmo

Para escribir un algoritmo deben seguirse un conjunto de pasos básicos

1. Comprender el problema
2. Identificar los elementos a incluir en el léxico: constantes, tipos, variables y acciones.
3. Encontrar la forma de secuenciar las acciones para obtener el resultado, esto es, alcanzar las poscondiciones a partir de un estado inicial que cumple con la precondición. Para establecer el orden de las acciones los lenguajes de programación proporcionan mecanismos de composición: Secuenciación, análisis de casos, iteración y recursion.
4. Al organizar las acciones en el tercer paso puede ocurrir que se detecte que faltan elementos en el léxico o que algún aspecto del problema no ha sido bien comprendido lo cual requeriría volver a los pasos anteriores.
5. Nunca la solución aparece en el primer intento, en general aparece en un proceso cíclico, entonces se debe:
6. Escribir el léxico,
    a. escribir la primera versión,
    b. incluir en el léxico nuevos elementos que faltaban,
    c. escribir la nueva versión del algoritmo y así sucesivamente

Estructura de un algoritmo

```
LEXICO {Léxico Global del algoritmo}
        {Declaración de tipos, constantes, variables y acciones}
        Acción 1
                PRE {Precondición de la acción 1}
                POS {Poscondición de la acción 1}
                LEXICO {Léxico local, propio de la acción 1}
                        Declaraciones locales
                ALGORITMO {Implementación de la acción 1}
                        {Secuencia de instrucciones de la acción 1}
                FIN {Fin implementación algoritmo de la acción 1}

ALGORITMO
        PRE {Precondición del algoritmo principal}
        POS {Poscondición del algoritmo principal}
                {Secuencia de instrucciones del algoritmo principal}
FIN {Fin del algoritmo principal}
```

Proceso Computacional

Se refiere a un algoritmo en ejecución. La ejecución de las instrucciones origina una serie de acciones sobre elementos de memoria que representan información manejada por el algoritmo. A nivel algorítmico se asigna un nombre a cada información de modo de manejar un par nombre-valor para cada información.

Una variable representa alguna entidad del mundo real, relevante para el problema que se quiere resolver. El efecto que producen las acciones del proceso sobre las variables produce cambio de estados o de sus valores.

Definiciones

Programa: Algoritmo escrito en un lenguaje cuyas instrucciones son ejecutables por una computadora y que están almacenados en un disco.
Tarea: Un programa se vuelve tarea a partir del momento que se lo selecciona para su ejecución y hasta que esta termina.
Proceso: programa en ejecución, se ha iniciado pero aún no ha finalizado.
Lenguajes de programación: notación que permite escribir programas a mayor nivel de abstracción que los lenguajes de máquina. Sus instrucciones deben ser traducidas a lenguaje de máquina.
Lenguaje de máquina: Instrucciones que son ejecutables por el hardware de una computadora.
Paradigmas de programación
Paradigma: Colección de conceptos que guían el proceso de construcción de un programa. Estos conceptos controlan la forma en que se piensan y formulan los programas.
Imperativo – Procedural – Objetos.
Declarativo – Funcional – Lógico.
Dato Información Conocimiento
Dato: <objeto><atributo><valor> sin interpretar.
Información: añade significado al dato.
Conocimiento: Añade propósito y capacidad a la información. Potencial para generar acciones.
Problema
Enunciado con una incógnita, la solución es encontrar el valor de esa incógnita.
Problema computacional o algorítmico: tarea ejecutada por una computadora con una especificación precisa de los datos de entrada y de los resultados requeridos en función de estos.
Clase de problemas
No computables: No existe un algoritmo.
Computables
Tratables: Existe un algoritmo eficiente.
Intratable: No existe algoritmo eficiente.
Expresiones Sentencias Léxico
Expresiones: secuencia de operadores y operandos que se reduce a un solo valor.
Sentencias: acción produce un efecto, puede ser primitiva o no primitiva.
Léxico: Descripción del conjunto de acciones e informaciones a partir de la cual se expresa el esquema de comportamiento del algoritmo.

Pasos para resolver un algoritmo
Comprender el problema.
Identificar información y acciones a incluir en el léxico (constantes, tipos, variables y acciones).
Encontrar un camino de secuenciar las acciones para obtener el resultado, es decir para alcanzar la poscondición a partir del estado inicial que cumple con la precondición.
Acciones primitivas y derivadas
Acciones primitivas: Incorporadas por el lenguaje.
Acciones derivadas: realizadas mediante la combinación de acciones primitivas con el objeto de desarrollar una tarea en particular. Son complementarias y pueden ser desarrolladas por el programador.

Estructura de un algoritmo

```
LEXICO {Léxico Global del algoritmo}
        {Declaración de tipos, constantes, variables y acciones}
        Acción 1
                PRE {Precondición de la acción 1}
                POS {Poscondición de la acción 1}
                LEXICO {Léxico local, propio de la acción 1}
                        Declaraciones locales
                ALGORITMO {Implementación de la acción 1}
                        {Secuencia de instrucciones de la acción 1}
                FIN {Fin implementación algoritmo de la acción 1}

ALGORITMO
        PRE {Precondición del algoritmo principal}
        POS {Poscondición del algoritmo principal}
                {Secuencia de instrucciones del algoritmo principal}
FIN {Fin del algoritmo principal}
```